API PAULINE

Spécifications partielles

Ce document vise à documenter les fonctions présentes dans les API Pauline. La liste des fonctions n'est pas exhaustive, étant donné que certaines informations ne sont pas encore disponibles. Cela concerne :

1. La gestion des extensions documentaires
2. La gestion des localisations dentaires
3. La gestion des notes héritées

Ces informations feront l'objet de tables spécifiques exploitables par des fonctions présentes dans la version définitive des API Pauline.

Les fonctions décrites dans ce document s'appuient sur des tables dont le format est décrit en annexe.

En cas de problème :

Les questions concernant les API Pauline pourront être adressées par courrier électronique à support@atih.sante.fr en précisant « API Pauline » dans l'objet du message ou par téléphone au 04 37 69 71 27 (de 9h à 12h).

`get_info_code()` **Donne les informations d'un acte**

Prototype

```
APIRET get_info_code(char dirtab[],char code[],code_info *info)
```

Appel

`dirtab` indique le répertoire contenant le fichier des actes CCAM
`code` indique le code dont on souhaite récupérer les informations
`info` ,de type code_info, est remplie avec les informations du code

Fonctionnement

Cette fonction a pour objet de fournir dans une structure de type code_info toutes les informations d'un acte donné

`get_libelle()` **Donne le libellé d'un acte**

Prototype

```
APIRET get_libelle(char dirtab[],char code[],char libelle[],long *offset)
```

Appel

`dirtab` indique le répertoire contenant le fichier des actes CCAM
`code` de l'acte dont on cherche le libellé
`libelle` retourne le libellé. Il doit être initialisé à '\0' par le programme appelant
`offset` précise éventuellement à quelle position la recherche doit commencer dans le fichier des actes CCAM. Lorsque le libellé est trouvé, offset est mis à jour et peut donc être utilisé par le programme appelant.

Fonctionnement

Cette fonction a pour objet de fournir le libellé d'un acte passé en paramètre.

Si le code n'existe pas, la fonction retourne ACTE_INCONNU.

`get_info_hier()` **Donne les informations d'une section hiérarchique**

Prototype

```
APIRET get_info_hier(char dirtab[],char hier[],hier_info *info)
```

Appel

`dirtab` indique le répertoire contenant le fichier des sections hiérarchiques
`hier` indique la section hiérarchique dont on souhaite récupérer les informations
`info` ,de type hier_info, est remplie avec les informations de la section hiérarchique

Fonctionnement

Cette fonction a pour objet de fournir dans une structure de type hier_info toutes les informations d'une section hiérarchique donnée

`get_ss_section()` **Recherche de sous sections**

Prototype

```
APIRET get_ss_section(char dirtab[],char motif[],int *nb_sous_sections,char **res,int position_section)
```

Appel

`dirtab` indique le répertoire contenant le fichier des sections hiérarchiques
`motif` est la section à rechercher :

- XX??0000 pour rechercher les sous chapitres
- XXXX??00 pour rechercher les paragraphes
- XXXXXX?? pour rechercher les sous paragraphes

`nb_sous_sections` renvoi le nombre de sous sections trouvées
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant
`position_section` précise la position de la sous section recherchée :

- 2 pour un sous chapitre
- 4 pour un paragraphe
- 6 pour un sous paragraphe

Fonctionnement

Cette fonction a pour objet de fournir en une liste toutes les sous sections de `motif`.

Si `motif` n'existe pas ou si `motif` n'a pas de sous section, `nb_sous_sections` retourne 0.

La liste en retour est une structure dont le premier élément indique la première sous section de la liste suivi des autres sous sections.

Organisation des résultats de get_ss_section()

ordre	type	contenu
1	char[position_section+2]	sous section n° 1
2	char[position_section+2]	sous section n° 2
…	…	…
nb_sous_sections	char[position_section+2]	sous section n° nb_sous_sections

`rech_hier()` **Recherche des actes dans une section terminale**

Prototype

```
APIRET rech_hier(char dirtab[],char motif[],int *nbcodes,char **res)
```

Appel

`dirtab` indique le répertoire contenant le fichier des actes CCAM
`motif` est la section à rechercher
`nbcodes` renvoi le nombre de codes trouvés
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant.

Fonctionnement

Cette fonction a pour objet de fournir en une liste tous les actes qui ont `motif` comme position hiérarchique. Cela suppose que motif soit une section terminale.

Si motif n'existe pas dans la CCAM ou si motif n'est pas une section terminale, nbcodes retourne 0.

La liste en retour est une structure dont le premier élément indique le premier acte de la liste suivi des autres actes.

Organisation des résultats de rech_hier()

ordre	type	contenu
1	char[7]	code de l'acte 1
2	char[7]	code de l'acte 2
...	...	...
nbcodes	char[7]	code de l'acte nbcodes

`rech_acte_racine()` **Recherche d'un acte sur la racine**

Prototype

```
APIRET rech_acte_racine(char dirtab[],char motif[],int *nbcodes,char **res)
```

Appel

`dirtab` indique le répertoire contenant le fichier des actes CCAM
`motif` est la racine à rechercher
`nbcodes` renvoi le nombre de codes trouvés
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant.

Fonctionnement

Cette fonction a pour objet de fournir en une liste tous les actes ayant motif pour racine. Le caractère joker « ? » peut être utilisé.

Si le code fourni n'existe pas dans la CCAM, nbcodes retourne 0.

La liste est une structure dont le premier élément indique le premier acte de la liste suivi des autres actes.

Organisation des résultats de rech_acte_racine()

ordre	type	contenu
1	char[7]	code de l'acte 1
2	char[7]	code de l'acte 2
...	...	...
nbcodes	char[7]	code de l'acte nbcodes

`rech_mot()` **Recherche d'un mot**

Prototype

```
APIRET rech_mot(char dirtab[],char motif[],int *nbcodes,char **res,int champ,int type_recherche)
```

Appel

`dirtab` indique le répertoire contenant le fichier des actes CCAM
`motif` est le mot à rechercher. Il doit être en majuscule non accentué
`nbcodes` renvoi le nombre de codes trouvés
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant.
`champ` indique dans quel champ du fichier la recherche doit s'effectuer
`type_recherche` indique le type de recherche, qui peut prendre les valeurs suivantes :

- RECHERCHE_FRAGMENT
- RECHERCHE_RACINE
- RECHERCHE_EXACTE

Fonctionnement

Cette fonction a pour objet de fournir en une liste tous les actes qui ont `motif` présent dans le `champ` passé en paramètre.

Le `champ` peut être le champ des libellés ou des notes d'utilisation de l'acte.

`motif` peut être recherché sur un fragment de chaîne, sur la racine d'une chaîne ou sur une chaîne exacte.

Si le code fourni n'existe pas dans la CCAM, nbcodes retourne 0.

La liste en retour est une structure dont le premier élément indique le premier acte de la liste suivi des autres actes.

Organisation des résultats de rech_mot ()

ordre	**type**	**contenu**
1	char[7]	code de l'acte 1
2	char[7]	code de l'acte 2
...	...	...
nbcodes	char[7]	code de l'acte nbcodes

`sous_recherche()` **Effectue une sous recherche**

Prototype

```
APIRET sous_recherche(char *res_courant,int nb_courant,char *res_nouvelle,int nb_nouvelle,
char **res,int *nbcodes)
```

Appel

`res_courant` contient la recherche courante
`nb_courant` contient le nombre d'éléments de la recherche courante
`res_nouvelle` contient la nouvelle recherche
`nb_nouvelle` contient le nombre d'éléments de la nouvelle recherche
`res` contient le résultat de la sous recherche dans res_courant de res_nouvelle
`nbcodes` contient le nombre d'éléments de la sous recherche dans res_courant de res_nouvelle

Fonctionnement

Cette fonction a pour objet de fournir en une liste une sous recherche.

La liste en retour est une structure dont le premier élément indique le premier acte de la liste suivi des autres actes.

Organisation des résultats de sous_recherche()

ordre	type	contenu
1	char[7]	code de l'acte 1
2	char[7]	code de l'acte 2
...	...	...
nbcodes	char[7]	code de l'acte nbcodes

`recherche_ou()` **Effectue une recherche « ou »**

Prototype

```
APIRET recherche_ou(char *res1,int nb1,char *res2,int nb2,char **res,int *nbcodes)
```

Appel

`res1` contient la première liste de résultats
`nb1` contient le nombre d'éléments de la première liste
`res2` contient la deuxième liste de résultats
`nb2` contient le nombre d'éléments de la deuxième liste
`res` renvoie le résultat de la recherche « ou » entre `res1` et `res2`
`nbcodes` renvoie le nombre de résultats de la recherche

Fonctionnement

Cette fonction a pour objet de fournir en une liste tous les actes d'une recherche « ou » entre deux listes d'actes passés en paramètres.

La liste en retour est une structure dont le premier élément indique le premier acte de la liste suivi des autres actes.

Organisation des résultats de recherche_ou()

ordre	type	contenu
1	char[7]	code de l'acte 1
2	char[7]	code de l'acte 2
...	...	...
nbcodes	char[7]	code de l'acte nbcodes

`liste_proc()` **Liste des actes constitutifs d'une procédure**

Prototype

```
APIRET liste_proc(char dirtab[],char code_in[],char phase,char activite,char **res)
```

Appel

`dirtab` indique le répertoire contenant le fichier de contrôle
`code_in` est le code de la procédure dont on veut lire la décomposition
`phase` est le code de la phase (« 0 » si sans objet)
`activite` est le code l'activité (« 0 » si sans objet)
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant.

Fonctionnement

Cette fonction a pour objet de fournir en une liste la décomposition d'une procédure, dont on fournit le code complété le cas échéant d'un code de phase, et éventuellement d'un code d'activité.

Si le code fourni n'existe pas dans la CCAM, une erreur est retournée. Quand la fonction retourne une erreur, le pointeur sur lequel pointe `res` est mis à `NULL`.

Si la procédure se décompose de plusieurs manières, la liste fournit chaque décomposition l'une après l'autre dans la structure `res` dont le premier élément est le nombre de décompositions distinctes. Cette structure est détaillée dans le tableau suivant :

Organisation des résultats de liste_proc()

ordre	type	contenu
1	int	ND : nombre de décompositions procédurales pour ce « code + phase + activité »
2	Int	N1 : nombre d'actes de la décomposition n°1
3	char[7],char,char	code acte, phase, activité de l'acte 1 de la décomposition n°1
4	char[7],char,char	code acte, phase, activité de l'acte 2 de la décomposition n°1
…	…	…
N1+1	char[7],char,char	code acte, phase, activité de l'acte N1 de la décomposition n°1
N1+3	int	N2 : nombre d'actes de la décomposition n°2
N1+4	char[7],char,char	code acte, phase, activité de l'acte 1 de la décomposition n°2
N1+5	char[7],char,char	code acte, phase, activité de l'acte 2 de la décomposition n°2
…	…	…

`liste_act_proc()` **Liste des procédures qui contiennent un acte donné**

Prototype

```
APIRET liste_act_proc(char dirtab[],char code_in[],char phase,char activite,char **res)
```

Appel

`dirtab` indique le répertoire contenant le fichier de contrôle
`code_in` est le code de l'acte pour lequel on recherche les procédures le comprenant
`phase` est le code de la phase (« 0 » si sans objet)
`activite` est le code l'activité (« 0 » si sans objet)
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant.

Fonctionnement

Cette fonction a pour objet de fournir la liste des procédures qui contiennent un acte donné complété le cas échéant d'un code de phase, et éventuellement d'un code d'activité.

Si le code fourni n'existe pas dans la CCAM, une erreur est retournée. Quand la fonction retourne une erreur, le pointeur sur lequel pointe `res` est mis à `NULL`.

Si un acte est présent dans plusieurs procédures, la liste fournit chaque procédure l'une après l'autre dans la structure `res` dont le premier élément est le nombre de procédures. Cette structure est détaillée dans le tableau suivant :

Organisation des résultats de liste_act_proc()

ordre	type	contenu
1	int	NP : nombre de procédures qui contiennent ce « code + phase + activité »
2	char[7],char,char	code acte, phase, activité de la procédure 1 qui contient cet acte
3	char[7],char,char	code acte, phase, activité de la procédure 2 qui contient cet acte
...	...	...
NP+1	char[7],char,char	code acte, phase, activité de la procédure NP qui contient cet acte

`liste_incomp()` **Liste des actes incompatibles avec un acte donné**

Prototype

```
APIRET liste_incomp(char dirtab[],char code_in[],char phase,char activite,char **res)
```

Appel

`dirtab` indique le répertoire contenant les fichiers de contrôle
`code_in` est le code de l'acte pour lequel on recherche les actes incompatibles
`phase` est le code de la phase (« 0 » si sans objet)
`activite` est le code l'activité (« 0 » si sans objet)
`res` est l'adresse d'un pointeur sur une structure, non instanciée par le programme appelant.

Fonctionnement

Cette fonction a pour objet de fournir en une liste toutes les séries d'incompatibilités relatives à un acte, dont on fournit le code, complété le cas échéant d'un code de phase, et éventuellement d'un code d'activité (chacune de ces deux informations étant fixée à « 0 » lorsqu'elle ne s'applique pas).

Si le code fourni n'existe pas dans la CCAM, une erreur est retournée. Quand la fonction retourne une erreur, le pointeur sur lequel pointe `res` est mis à `NULL`.

Cette fonction fournit la composition de chaque combinaison incompatible dans laquelle l'acte (un code, une phase et une activité donnés) est impliqué. Chaque incompatibilité est documentée, c'est-à-dire que la raison de l'incompatibilité est indiquée dans la liste en retour, dont le premier élément est le nombre de combinaisons incompatibles impliquant cet acte.

Organisation des résultats de liste_incomp()

ordre	type	contenu
1	int	NC : nombre de combinaisons incompatibles impliquant ce « code + phase + activité »
2	char	Type d'incompatibilité de la combinaison n°1
3	char[7],char,char	code acte, phase, activité de l'acte 1 de la combinaison n°1
4	char[7],char,char	code acte, phase, activité de l'acte 2 de la combinaison n°1
5	char	Type d'incompatibilité de la combinaison n°2
6	char[7],char,char	code acte, phase, activité de l'acte 1 de la combinaison n°2
7	char[7],char,char	code acte, phase, activité de l'acte 2 de la combinaison n°2
...	...	...

Structures principales

Extrait du fichier ***api_pauline_types.h*** :

```
/* Retour des fonctions */
typedef enum
{
  OK                    = 0,   /* Traitement sans probleme                  */
  PB_OUVERTURE_CCAM_ACTES,     /* Probleme d'ouverture du fichier CCAM_ACTES */
  PB_OUVERTURE_CCAM_HIER,      /* Probleme d'ouverture du fichier CCAM_HIER  */
  PB_OUVERTURE_CCAM_CTRL,      /* Probleme d'ouverture du fichier CCAM_CTRL  */
  ACTE_INCONNU,                /* Acte inconnu                              */
  HIER_INCONNU,                /* Hierarchie inconnue                       */
  PB_MEMOIRE            = 9    /* Probleme memoire                          */
} APIRET;

/* Activite d'un acte */
typedef struct
{
  char num_activite;
  char libelle_activite[TAILLE_LIBELLE];
  char gestes_compl[TAILLE_LIBELLE];
  char modificateurs[TAILLE_LIBELLE];
} activite_info;

/* Phase d'un acte */
typedef struct
{
  char libelle_phase[TAILLE_LIBELLE];
  int nb_activites_phase;
  activite_info activites_phase[MAX_ACTIVITES];
} phase_info;

/* Information d'un code */
typedef struct
{
  int nb_phases;
  phase_info phases[MAX_PHASES];
  int nb_activites;
  activite_info activites[MAX_ACTIVITES];
  char position_hier[TAILLE_HIER+1];
  char sexe;
  char prise_charge;
  char classant;
```

```
  char note[TAILLE_NOTE];
  char libelle[TAILLE_LIBELLE];
} code_info;

/* Information d'une section hierarchique */
typedef struct
{
  char libelle[TAILLE_LIBELLE];
  char note[TAILLE_NOTE];
} hier_info;
```

Format des tables

FICHIER CCAM_ACTES

Cette table sert à stocker les informations sur les actes.

N° du champ	Intitulé	Longueur
1	Code CCAM	7
2	Phase	1
3	Activité	1
4	Position hiérarchique	8
5	Sexe compatible	1
6	Prise en charge exceptionnelle	1
7	Caractère classant	1
8	Gestes complémentaires	Variable
9	Modificateurs	Variable
10	Libellé et note de la phase	Variable
11	Libellé de l'activité	Variable
12	Note d'utilisation de l'acte	Variable
13	Libellé de l'acte	Variable

FICHIER CCAM_POSHIER

Cette table sert à stocker les informations concernant les différentes sections hiérarchiques de la CCAM.

N° du champ	Intitulé	Longueur
1	Position hiérarchique	8
2	Index	4
3	Libellé de la section ou de la note	Variable

L'index peut prendre les valeurs suivantes :

- 0000 : indique que l'on est au niveau d'un libellé de section
- 0001 à XXX : indique l'index unique de la note

FICHIER CCAM_CTRL

Cette table contient toutes les combinaisons qui peuvent faire l'objet d'un contrôle à la saisie :

- décomposition des procédures en actes isolés
- actes incompatibles entre eux

Les champs de cette table sont récapitulés dans le tableau suivant :

Numéro du champ	Intitulé	Longueur
1	Nature de la combinaison	2
2	Numéro optionnel N	2
3	Premier code CCAM C1	7
4	Premier code phase P1	1
5	Premier code activité A1	1
6	Second code CCAM C2	7
7	Second code phase P2	1
8	Second code activité A2	1

Nature de la combinaison :

01 : les procédures sont des combinaisons d'actes de la CCAM, dont la réalisation simultanée est suffisamment fréquente pour qu'on ait estimé justifié de donner un code et un libellé pour leur association. Il est obligatoire de coder la procédure et non pas ses actes constitutifs, dès lors qu'ils ont été réalisés simultanément. De même, une procédure et l'un de ses actes constitutifs ne peuvent être codés simultanément. A noter : une procédure peut elle-même être l'un des constituants d'une autre procédure.

02 : les actes incompatibles constitutifs d'« associations interdites » hormis le cas des procédures, qui représentent des actes « très fréquemment associés », certains actes ne peuvent être codés simultanément, en général pour des raisons exactement inverses : impossibilité matérielle, éthique, technique, etc. de réaliser simultanément deux ou plusieurs actes.

Numéro optionnel N

Pour une décomposition de procédure, ce complément optionnel permet de distinguer les différentes décompositions possibles d'une même procédure : l'ensemble des enregistrements qui comportent le même numéro optionnel N et le même « premier » code CCAM décrivent (en second code) les éléments de l'une des décompositions possibles de la procédure codée en premier code. On peut ainsi concevoir ce complément comme une numérotation des décompositions possibles.

Pour une combinaison d'actes incompatibles, ce champ précise la raison de l'incompatibilité. S'il vaut 00, la raison de l'incompatibilité n'est pas précisée.

Premier code CCAM (C1)

Pour une décomposition de procédure, il s'agit du code CCAM de la procédure en question.

Pour une incompatibilité, il s'agit d'un premier code CCAM dont l'association avec le second code CCAM (C2) est interdite.

Premier code phase (P1)

Ce champ permet le cas échéant de limiter l'application de la combinaison décrite à une seule des phases de l'acte codé C1 (au cas, bien entendu, où l'acte C1 dispose de phases). Si ce champ est laissé à blanc, ou vaut 0, cela signifie que le code C1 est pris en compte pour la détermination de la combinaison indépendamment de la phase (en d'autres termes, pour un acte C1 comportant plusieurs phases avec P1 à blanc, toutes les phases sont concernées par la combinaison impliquant C1).

Premier code activité (A1)

De la même manière que pour le code phase (P1), A1 permet de restreindre la combinaison impliquant l'acte codé C1 à une seule de ses activités. Si ce champ est laissé à blanc, ou vaut 0, cela signifie que le code C1 est pris en compte indépendamment de l'activité (toutes les activités du code C1 sont concernées par la combinaison impliquant C1).

Second code CCAM (C2)

Pour une décomposition de procédure, il s'agit du code CCAM d'un des actes isolés compris dans cette décomposition de la procédure. L'ensemble des codes indiqués dans ce champ pour des enregistrements relatifs à une même décomposition (champ premier code C1 identique ET champ numéro optionnel N identique) constituent les éléments de la procédure.

Pour une incompatibilité, il s'agit du second code CCAM qui est incompatible avec le code C1.

Second code phase (P2)

Ce champ permet le cas échéant de limiter l'application de la combinaison décrite à une seule des phases de l'acte codé C2 (au cas, bien entendu, où l'acte C2 dispose de phases). Si ce champ est laissé à blanc, ou vaut 0, cela signifie que le code C2 est pris en compte pour la détermination de la combinaison indépendamment de la phase (en d'autres termes, pour un acte C2 comportant plusieurs phases avec P2 à blanc, toutes les phases sont concernées par la combinaison impliquant C2).

Second code activité (A2)

De la même manière que pour le code phase (P2), A2 permet de restreindre la combinaison impliquant l'acte codé C2 à une seule de ses activités. Si ce champ est laissé à blanc, ou vaut 0, cela signifie que le code C2 est pris en compte indépendamment de l'activité (toutes les activités du code C2 sont concernées par la combinaison impliquant C2).